# Visualizing High-Resolution Accelerator Physics

Patrick S. McCormick
Advanced Computing Lab, Los Alamos National Laboratory

Robert D. Ryne
Accelerator Physics and Engineering, Los Alamos National Laboratory

## 1. INTRODUCTION

Particle accelerators are playing an increasingly important role in basic and applied science. Several countries are involved in efforts aimed at developing accelerator-driven technologies in support of several different application domains, including high energy and nuclear physics, material science, biological science, and military use. The technological challenges associated with designing the next-generation of accelerators will require numerical modeling capabilities that are far beyond those normally used within the accelerator community. For example, future high-average-power linear accelerators will have to operate with extremely low beam loss (of the order 0.1nA/m) to prevent unacceptably high levels of radioactivity. To ensure this requirement is met, it is necessary to perform very high-resolution simulations using hundreds of millions to billions of particles in which the beam propagates through kilometers of complicated accelerating structures. These calculations require computational performance of hundreds of GFLOPS to TFLOPS and core memory requirements of hundreds of Gbytes.

In 1997 the U.S. Department of Energy initiated a Grand Challenge in Computational Accelerator Physics. The primary goal of this project is to develop a new generation of high-performance accelerator modeling tools and apply them to projects of national importance. The development of these tools will have a major impact on reducing the cost and technical risk of future projects, as well as maximizing the performance of present and future accelerators. In addition, this will enable the simulation of problems three to four orders of magnitude larger than has ever been done before. These simulations are capable of resolving features critical to the understanding of accelerator beam dynamics. The use of algorithms and software optimized for high-performance computing will make it possible to obtain results quickly and with very high accuracy. This work is being done in collaboration between Los Alamos National Laboratory (LANL), Stanford University, National Energy Research Scientific Computing Center, and the University of California at Los Angeles. The remainder of this article focuses on the accelerator simulation model and the current techniques used to visualize the results produced by the project.

## 2. BEAM DYNAMICS SIMULATIONS

The most widely used approach for modeling the dynamics of intense charged particle beams is based on particle simulation techniques. The particles themselves are subject to two types of forces: the external forces used to guide, focus, and accelerate the beam, and the self-forces (called space charge forces) of the particles interacting with one another. Accelerators typically have $10^9$-$10^{13}$ particles per bunch. It is common to perform simulations with tens to hundreds of millions of particles and in larger, less frequent runs, billions of particles have been used. The common output from a single time step of a simulation are the spatial coordinates of each particle $(x, y, z)$, and the momenta of each particle $(p_x, p_y, p_z)$. The space of coordinates and momenta is called phase space. The goal of beam dynamics simulations is to understand the evolution of the beam in phase space. This is the mathematical analog of the beam's evolution as it propagates along the accelerator. Two codes are currently producing these results: one using Fortran 90 [**?**; **?**] and the other using the POOMA C++ framework [**?**]. One of the main goals of the simulations is to understand the motion of particles in the beam halo, the very low density region of charge far from the beam core, that is responsible for the interaction of particles with the accelerating structures. Understanding and predicting the beam halo is a major issue for next-generation, high-current accelerators. Given the large number of particles present in each time step, the results are challenging to visualize at interactive rates. The following sections discuss the techniques currently in use for the visualization of the results.

## 3. PARTICLE VISUALIZATION

In the past, simulated particle data has been visualized by using a batch process that extracted two-dimensional slices through the particle field. The resulting slices are then viewed as either individual particles or as a particle density distribution. Figure 1 shows an example of this technique. While this approach provides reasonable results it has limitations. In particular, batch extraction and rendering require one or more fixed parameters that often make it difficult to study different aspects of the data. The two-dimensional nature of the results also make it difficult to understand the relationship between particles in different slices. Full three-dimensional representations of the data have been used as well. The first approach uses hardware accelerated OpenGL rendering of either point or polygonal based representations of the individual particles. Even though this approach provides a true three-dimensional representation of the particles, the number of particles being simulated make it expensive and prohibitive for interactive exploration of the data. Benchmarks have shown that it takes approximately 50 seconds to render three-hundred million points on a Silicon Graphics InfiniteReality (IR) engine. While data reduction techniques are one viable solution for achieving interactive rendering rates, such techniques are often limiting in terms of studying the full details of the simulation. For example, down sampling of the particle data without concern for the structure of the beam can have a large impact on the details seen in the beam's halo. Therefore, the effects of a particular reduction technique must be well understood before it can be successfully used. Finally, past work at LANL's Advanced Computing Lab has looked at parallel software rendering of particle data

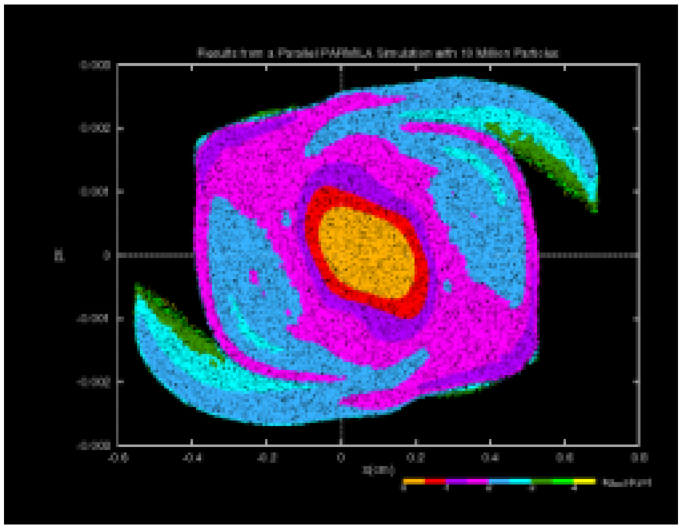

Fig. 1. Figure 1. Results showing a two-dimensional slice through the horizontal phase $(x, p_x)$ space of ten million particles. The colors in the image represent particle density.

(spheres) [?]. Although this approach can render hundreds of thousands of spheres per second it is not fast enough to interactively explore the data.

The most recent and successful approach for visualizing the beam dynamics is based upon volume rendering of the density distribution of the particles. A particle density volume may be created by determining the bounds of the particle distribution and then mapping individual particles into the voxels of a volume. After this process, each voxel contains a count of the number of particles that have been mapped to it. This density representation can then be visualized using hardware accelerated volume rendering based upon the OpenGL three-dimensional texturing extensions available on the IR engine [?; ?]. This approach provides rendering rates of 5-10 frames per second for $256^3$ volumes. In addition to supporting *semi-interactive* rendering rates, this technique also provides interactive editing of the transfer function which allows the user to selectively control the transparency and colors used in rendering. This is implemented by storing the density volume as 8-bit values that are used to index into the hardware's texture lookup table (TLUT); thus avoiding reloading the entire volume into texture memory. Therefore, only a 256 entry TLUT needs to be modified as the user edits the transfer function. The ability to interactively modify colors and transparency has proven to be a powerful technique for studying the details of the halo formation. Logarithmic mapping, data histograms, and trial-and-error techniques are used to establish reasonable transfer functions.

The determination of the density volume's size is influenced by two factors. The first is the number of particles that have been simulated, and the second is based upon the desired level of interactivity. It is important to note that each dimension of the volume is restricted, by the hardware, to be a power of two. A maximally configured IR pipe has enough texture memory (64MB) to store a $512 \times 512 \times 256$ byte volume. This limitation can introduce artifacts as the number of simulated particles grows beyond the voxel resolution. In this case, the spatial resolution of the data is reduced and it becomes difficult to study the features of the data. Problems also occur when the number of voxels is much larger than the number of particles. In this case, the dynamic range of the data is reduced, making it very difficult to explore the data.

In order to support larger volumes there are several different approaches that can be used. One possibility is to make multiple passes by rendering subsets of the volume during each pass. While solving the problem of rendering volumes larger than texture memory, this approach introduces a performance penalty due to the multiple passes. A second approach is to increase the amount of available texture memory by using more than one IR pipe for rendering.

## 4. PARALLEL RENDERING

By using software developed in collaboration with Silicon Graphics, and an Origin 2000 containing multiple IR pipes, volumes larger than the 64MB single pipe limit, can be rendered in parallel. Parallel rendering of a volume is achieved by subdividing the volume into sub-volumes and assigning each sub-volume to an individual pipe. The resulting image from each pipe is then passed on to the next pipe in the chain, where it is composited with the local result. A simplified view of this architecture is shown in Figure 2. This configuration allows nearly the same semi-interactive rendering rates of the single pipe approach. The supported size of the data set is then dependent upon the number of pipes in use. The available texture memory grows linearly as pipes are added. For example, two pipes can store a $512^3$ byte volume. By using 16 IR pipes, volumes as large as $1024^3$ have been visualized using this technique.

The main disadvantage of this approach is the latency introduced by the multi-pipe compositing structure; since each pipe must wait for the results of the previous pipe. For a small number of pipes this latency is acceptable; however, higher numbers of pipes (e.g. 8 or 16) introduce a latency that causes unacceptable delays in responding to user requests.

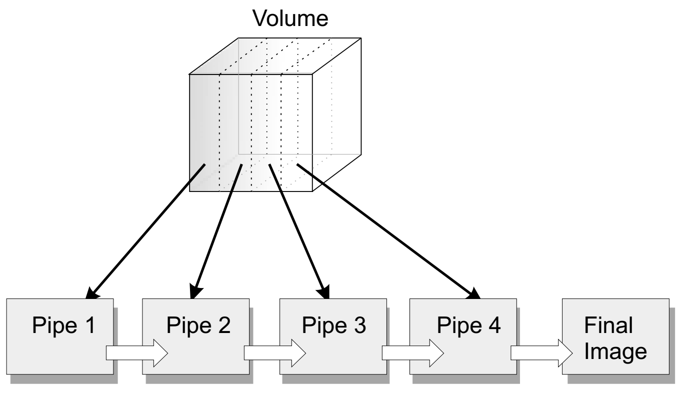

Fig. 2. Parallel hardware rendering configuration.

## 5. RESULTS

The images shown in Figures 3 and 4 present the results produced by the volume rendering of particle densities. Both figures are based upon a single time step from a three hundred million particle simulation mapped into a $512^3$ volume. Figure 3 presents a view of the spatial coordinate representation with half of the particle data cut away. The blue, magenta and yellow colors represent the dense core of the particle beam. Of particular interest in this figure is the low density region, shown as a transparent red surrounding the data. This is the halo feature of the beam. Figure 4 shows the phase space $(x, p_x, z)$ representation of the data. In this case, the halo is shown by the low density *arms* that extend over the front and back of the dark cyan colored, high density, region of the core. The arms are colored, in the order of decreasing density, light cyan, white, yellow and reddish-orange. In both images the transfer function editor was used to highlight the features of interest – in this case the low density regions.

The true three-dimensional nature of the particle density representations are often very difficult to resolve; the eye tends to flatten them out into two dimensions. Simple techniques, like the interactive rotation of the data and carefully choosing colors and transparency values can help reduce this effect. Adding stereo rendering support to the volume renderer has proved to be an invaluable technique for understanding the three-dimensional structure of the beam. Unfortunately, this technique reduces rendering performance by a factor of two.

## 6. FUTURE WORK

The results presented above are very qualitative and do not answer many of the detailed, quantitative questions required to provide a complete understanding of the data. Future work will focus on the development of query-based transfer function editing, the addition of annotations (axes, colorbars, etc), and interactive data probes. The choice of a transfer function plays a very important role in the understanding of beam halo features. Therefore, future work is planned to develop new tools and better techniques for selecting and editing transfer functions. Additional studies of effective mappings from particles to density volumes are also of interest. For example, individual particles might be mapped such that they effect a neighborhood of voxels. Techniques like this may help resolve some of the artifact issues disscussed above.

Important future additions for the volume renderer include the ability to render embedded geometries as well as shading to enhance the three-dimensional nature of the data. Finally, work is currently in progress to improve upon the presented multi-pipe rendering architecture by reducing latency and optimizing overall performance.

## 7. ACKNOWLEDGMENTS

Special thanks go to Javier Castellar and Silicon Graphics for the their work on the initial implementation of the multi-pipe volume renderer. Thanks to the members of the Visualization Team at LANL's Advanced Computing Lab for their advice and support: James Ahrens, Alan Keahey, Allen McPherson, Jamie Painter and Cathe Ragsdale. We acknowledge the Advanced Computing Laboratory of Los Alamos National Laboratory. This work was performed on computing resources located at

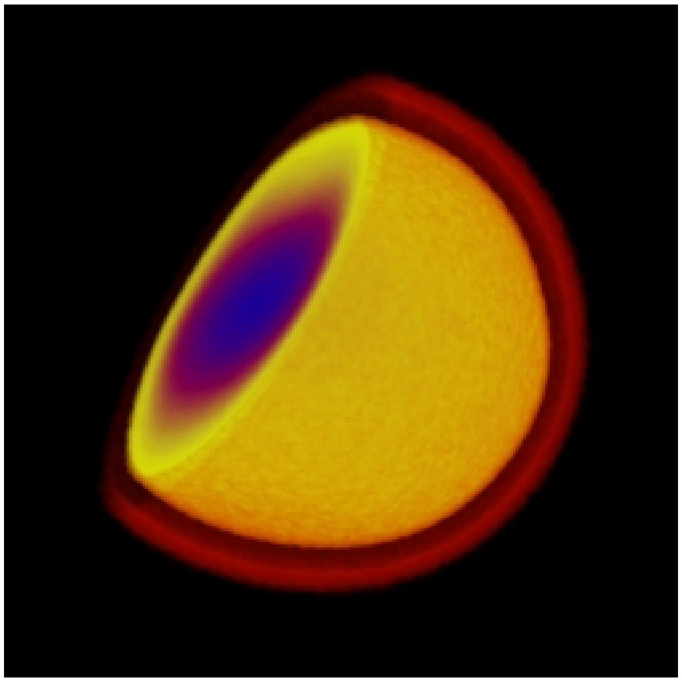

Fig. 3. Spatial coordinate $(x, y, z)$ view of volume rendered particle density. Based on a $512^3$ mapping of 300 million particles.

this facility.

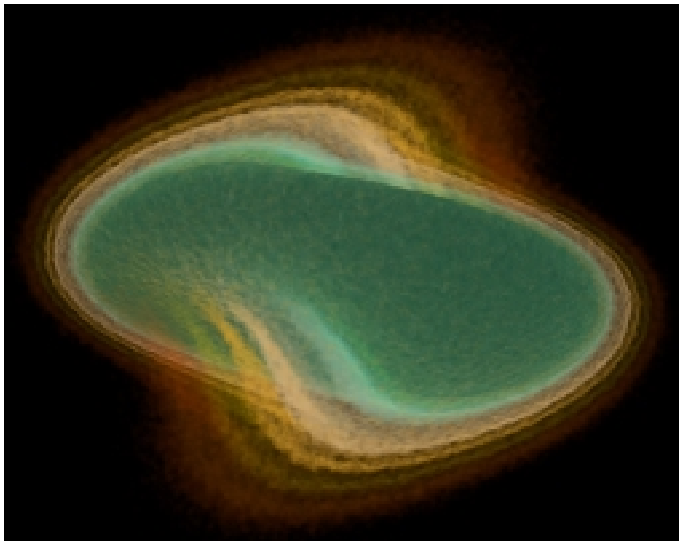

Fig. 4. Phase space $(x, p_x, z)$ view of volume rendered particle density. Based on a $512^3$ mapping of 300 million particles.